###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«ИЗМЕРЕНИЕ СТЕПЕНИ АССОЦИАТИВНОСТИ КЭШ-ПАМЯТИ»

студента 2 курса, группы 21203

Неретина Степана Ивановича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кандидат технических наук, доцент кафедры параллельных вычислений

А.Ю. Власенко

Новосибирск 2020

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#__RefHeading___Toc18443921)

[ЗАДАНИЕ 4](#__RefHeading___Toc18443922)

[ОПИСАНИЕ РАБОТЫ 5](#__RefHeading___Toc18443923)

[ЗАКЛЮЧЕНИЕ 15](#__RefHeading___Toc18443924)

[Приложение 1. Код программы 6](#__RefHeading___Toc18443925)

[Приложение 2. J](#__RefHeading___Toc18443925)SON вывода данных  [8](#__RefHeading___Toc18443925)

Приложение 3. Информация о процессоре  [9](#__RefHeading___Toc18443925)

Приложение 4. Графики при различных значениях «Offset» ……………………………….10

Приложение 5. Python скрипты для рисования графиков. …………………………………. 13

# **ЦЕЛЬ**

Экспериментальное определение степени ассоциативности кэш-памяти

# **Задание**

1. Написать программу, выполняющую обход памяти в соответствии с заданием

2. Измерить среднее время доступа к одному элементу массива (в тактах процессора) для разного числа фрагментов: от 1 до 32. Построить график зависимости времени от числа фрагментов.

3. По полученному графику определить степень ассоциативности кэш памяти, сравнить с реальными характеристиками исследуемого процессора.

4. Составить отчет по лабораторной работе(Следующий график показывает зависимость времени обращения к одному элементу в тактах от количества фрагментов в массиве).

# ОПИСАНИЕ РАБОТЫ

1. **График роста тактов обращения**

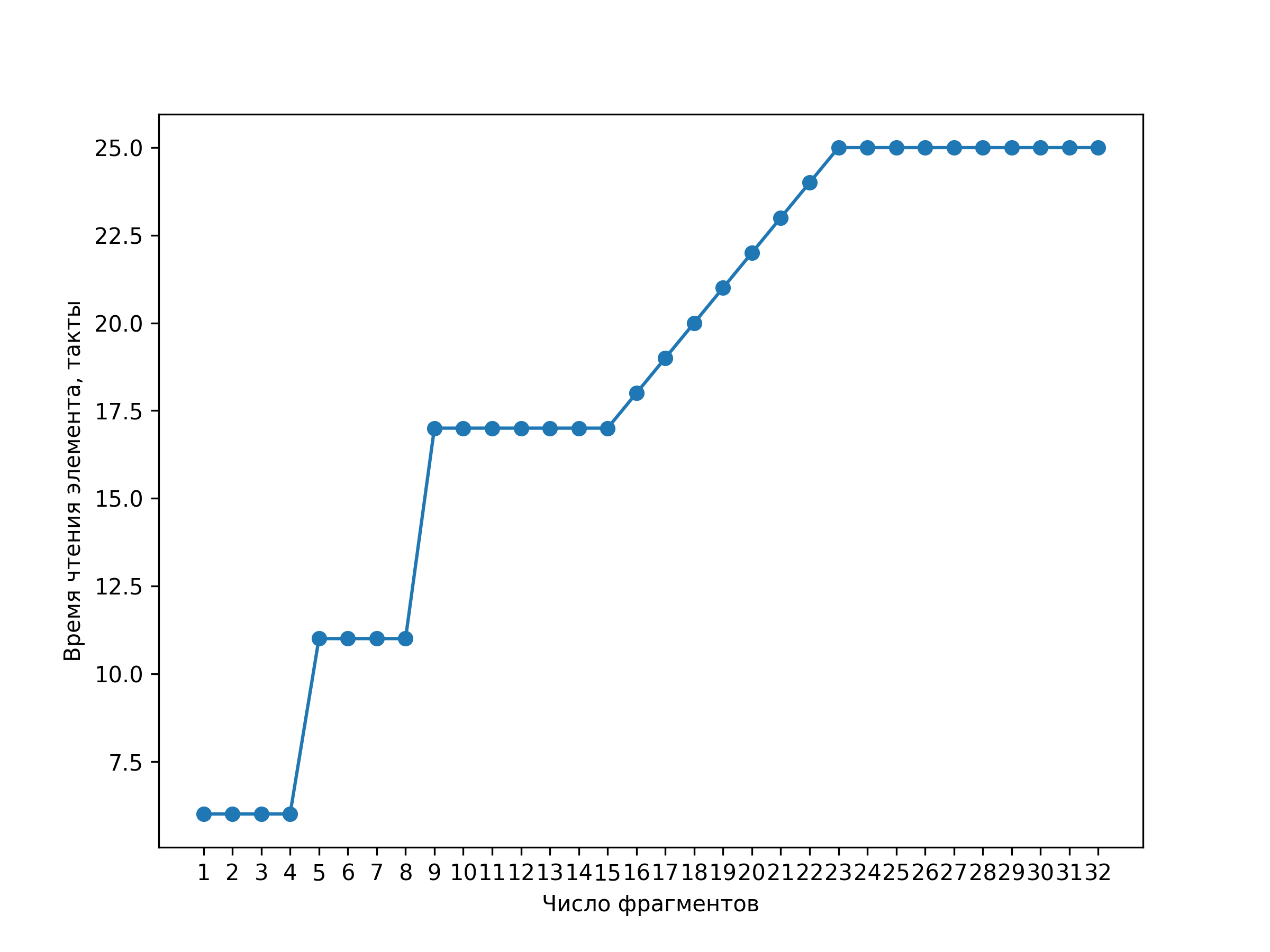
1) Написан код, который реализует поставленную задачу. Заполнение массивов производилось следующим образом:

2) Программа скомпилирована средствами IDE CLion.

3) Внутри программы создан цикл для многократной прогонки на каждом значений фрагментов. Тестирования проводилось без вмешательства, параллельно на компьютере ничего не выполнялось.

4) В результате полученные следующие данные:

На графике заметно два скачка. Первый скачок происходит после 4 фрагментов, что, скорее всего, говорит о степени ассоциативности L1. Следующий скачок роста времени обращения произошел на 8-ом фрагменте, что соответствует ассоциативности L2. Следующий скачок роста времени обращения произошел на 16-ом фрагменте, что соответствует ассоциативности L3.



# **Приложение 1 . Код программы**

#include <iostream>

#include <climits>

#include <random>

#include <cstring>

using namespace std;

const int RUN\_TIMES = 50;

const inline int OFFSET = 32\*1024;

const inline int FRAGMENTS\_NUM = 32;

const inline int N = OFFSET\*FRAGMENTS\_NUM;

constexpr inline int MATRIX\_N = 1024;

inline float generateRandomInRange(float a, float b) {

random\_device rd;

uniform\_real\_distribution gen(a, b);

return gen(rd);

}

float \*GenerateMatrix() {

auto \*A = new float[MATRIX\_N \* MATRIX\_N];

for (size\_t i = 0; i < MATRIX\_N; i++) {

for (size\_t j = 0; j < MATRIX\_N; j++) {

A[i \* MATRIX\_N + j] = generateRandomInRange(1, 100000);

}

}

return A;

}

float \*MultiplyMatrices(const float \*M1, const float \*M2) {

auto\* M3 = new float[MATRIX\_N \* MATRIX\_N];

memset(M3, 0, sizeof(float) \* MATRIX\_N \* MATRIX\_N);

for (size\_t i = 0; i < MATRIX\_N; ++i) {

float\* c = M3 + i \* MATRIX\_N;

for (size\_t k = 0; k < MATRIX\_N; ++k) {

const float\* b = M2 + k \* MATRIX\_N;

float a = M1[i \* MATRIX\_N + k];

for (size\_t j = 0; j < MATRIX\_N; ++j){

c[j] += a \* b[j];

}

}

}

return M3;

}

void init\_array(int \*arr, const size\_t fragments) {

for (size\_t i = 0; i < N - OFFSET \* (fragments - 1) ; i++) {

for (size\_t j = 0; j < fragments - 1; j++) {

//arr[i + (j - 1)\*OFFSET] = static\_cast<int>(i + j\*OFFSET);

arr[i + j\*OFFSET] = static\_cast<int>(i + (j + 1) \* OFFSET);

}

arr[i + OFFSET \* (fragments - 1)] = i + 1;

}

}

unsigned long long runLoop(int \*arr, size\_t fragments) {

init\_array(arr, fragments);

auto minTime = ULLONG\_MAX;

for (size\_t iterations\_count = 0; iterations\_count < RUN\_TIMES; iterations\_count++) {

auto start = \_\_builtin\_ia32\_rdtsc();

for (volatile size\_t k = 0, i = 0; i < N; i++) {

k = arr[k];

}

auto end = \_\_builtin\_ia32\_rdtsc();

if (minTime > end - start) {

minTime = end - start;

}

}

return minTime;

}

void countTime(int \*arr, int fragments){

init\_array(arr, fragments);

std::cout << fragments << " " << runLoop(arr, fragments) / N << std::endl;

}

int main() {

auto \*matrix\_1 = GenerateMatrix();

auto \*matrix\_2 = GenerateMatrix();

auto \*matrix\_3 = MultiplyMatrices(matrix\_1, matrix\_2);

if(matrix\_3[0] == 12345689){

printf("wow");

}

auto \*array = new int[N];

for (size\_t fragments = 1; fragments <= N/OFFSET; fragments++) {

countTime(array, static\_cast<int>(fragments));

}

delete[] array;

return 0;

}

# **Приложение 2 . Json данные вывода**

Data.json

[

[

1,

6

],

[

2,

6

],

[

3,

6

],

[

4,

6

],

[

5,

11

],

[

6,

11

],

[

7,

11

],

[

8,

11

],

[

9,

17

],

[

10,

17

],

[

11,

17

],

[

12,

17

],

[

13,

17

],

[

14,

17

],

[

15,

17

],

[

16,

18

],

[

17,

19

],

[

18,

20

],

[

19,

21

],

[

20,

22

],

[

21,

23

],

[

22,

24

],

[

23,

25

],

[

24,

25

],

[

25,

25

],

[

26,

25

],

[

27,

25

],

[

28,

25

],

[

29,

25

],

[

30,

25

],

[

31,

25

],

[

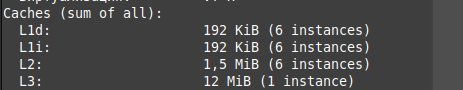
32,

25

]

]

# **Приложение 3. Информация о процессоре**

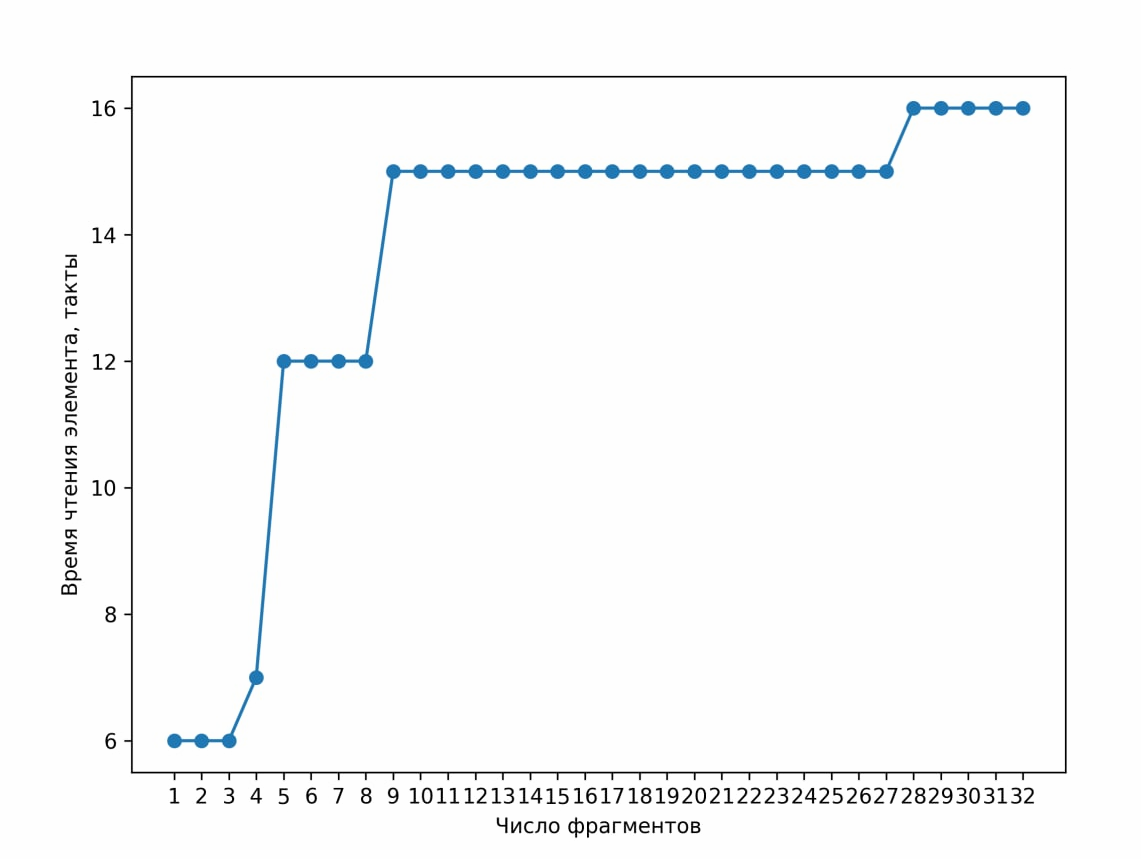
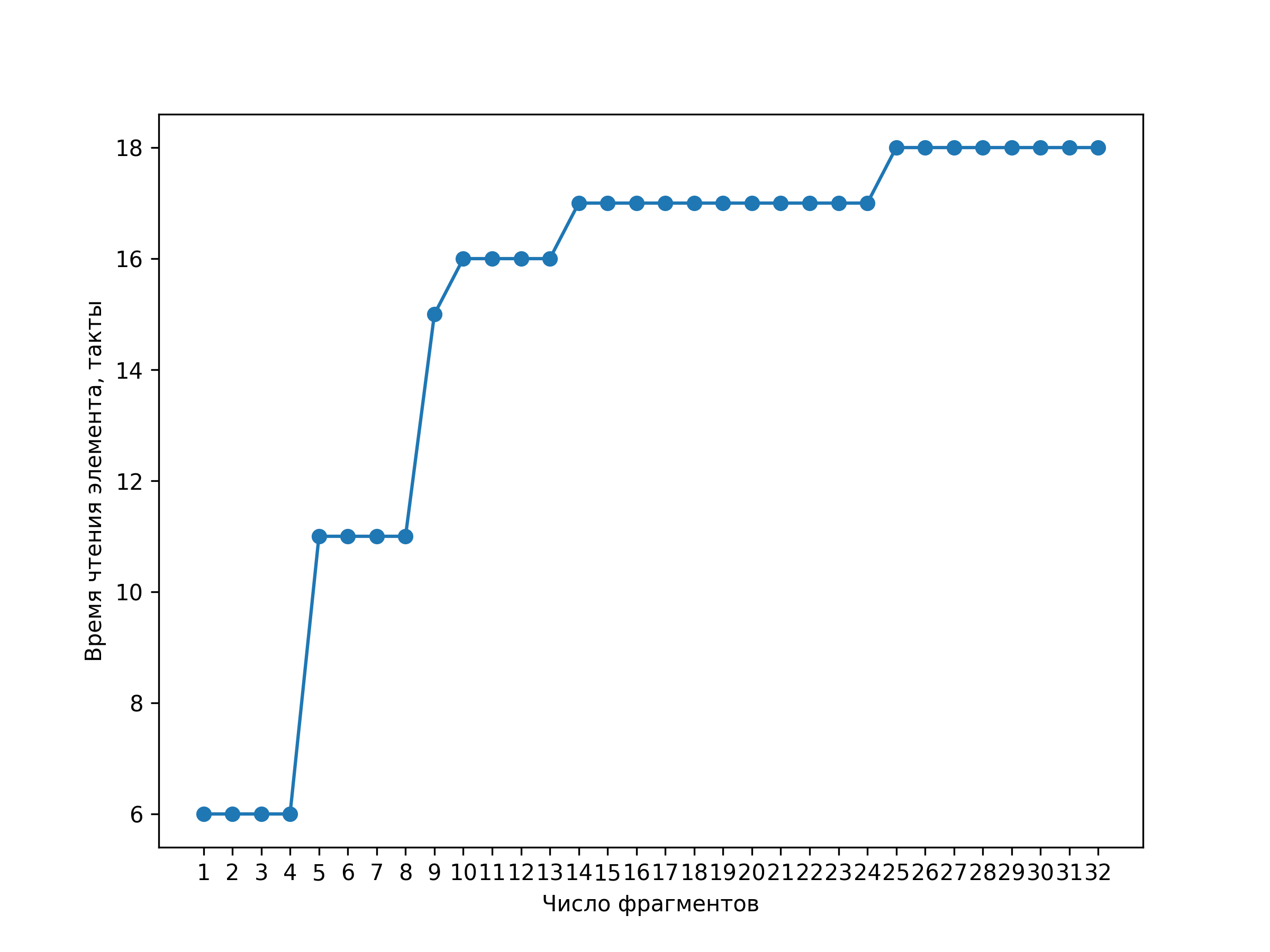


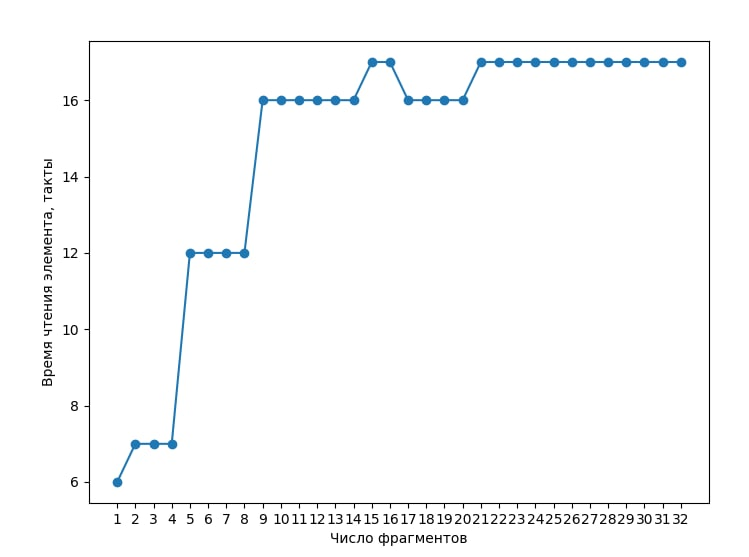
Информация об ассоциативности, полученная из конфигураций:



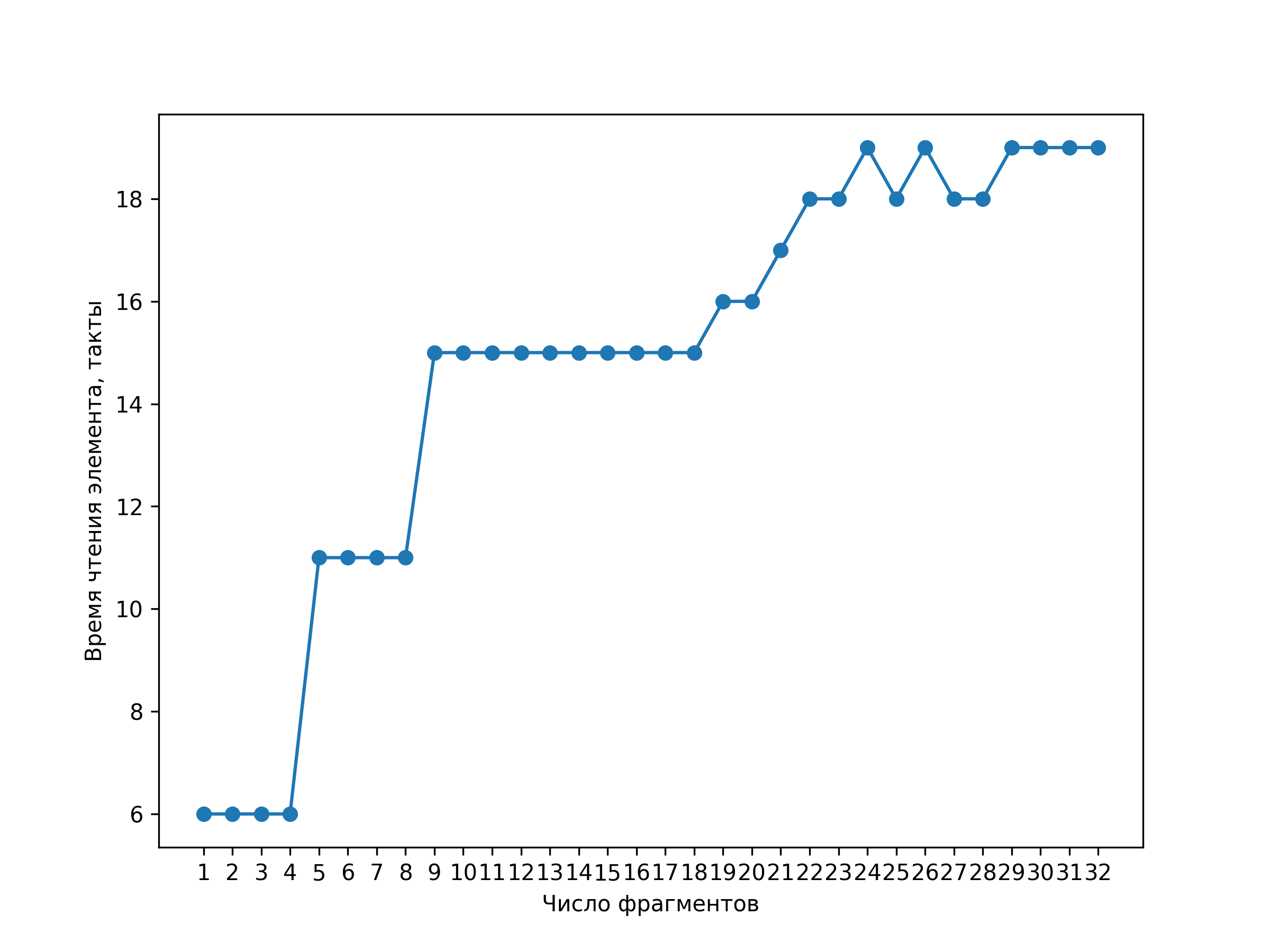
# **Приложение 4. Графики при различных значениях «Offset»**

**L1**

****

**L2**

**L3**

****

# **Приложение 5. Python скрипты для рисования графиков.**

Скрипт для сбора данных в JSON:

import json

import subprocess

for i in range(1, 100):

out = subprocess.Popen(['./cmake-build-debug/lab6'], stdout=subprocess.PIPE)

data = out.stdout.readlines()

try:

with open('kek/data' + str(i) + '.json', 'w') as f:

res = []

for item in data:

line = item.strip().decode("utf-8").split(" ")

x = int(line[0])

y = int(line[1])

res.append([x, y])

f.write(json.dumps(res))

except Exception as e:

print(e)

Скрипт для рисования графиков

from matplotlib import pyplot as plt

import json

plt.figure(figsize=(8, 6))

data = json.load(open("data.json"))

data = sorted(data, key=lambda el: el[0])

x = list([a[0] for a in data])

y = list([a[1] for a in data])

plt.plot(x,y)

plt.scatter(x, y)

xticks = []

plt.xticks(x)

plt.xlabel("Число фрагментов")

plt.ylabel("Время чтения элемента, такты")

plt.savefig('res1.png', dpi=300)

plt.show()

**Заключение**

В ходе данной работы было изучено строение множественно-ассоциативного кэша и эксперементально получены степени ассоциативности кэш памяти первого, второго и третьего уровня.